

DTIC FILE COPY

1

AD-A203 051



REAL-TIME DISPLAY OF TIME DEPENDENT DATA  
USING A HEAD-MOUNTED DISPLAY

THESIS

Gary Kim Lorimor  
Captain, USAF

AFIT/GE/ENG/88D-22

DTIC  
ELECTE  
JAN 18 1989

C&D

**DISTRIBUTION STATEMENT A**  
Approved for public release  
Distribution Unlimited

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

89

1 17 11

AFIT/GE/ENG/88D-22

DTIC  
ELECT  
JAN 18 1989  
S D<sup>6</sup>

REAL-TIME DISPLAY OF TIME DEPENDENT DATA  
USING A HEAD-MOUNTED DISPLAY

THESIS

Gary Kim Lorimor  
Captain, USAF

AFIT/GE/ENG/88D-22

Approved for public release; distribution unlimited

AFIT/GE/ENG/88D-22

REAL-TIME DISPLAY OF TIME DEPENDENT DATA  
USING A HEAD-MOUNTED DISPLAY

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology  
Air University  
In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Computer Engineering

Gary Kim Lorimor, B.S.  
Captain, USAF

December, 1988



ACCESSION FOR	
NTIS	CHART ✓
DTIC	TAB 51
Unpublished	
Justified	
By	
District	
Reference	
Dist	A-1

Approved for public release; distribution unlimited

## *Preface*

I would like to thank all those who made this thesis effort possible. Dad and Mom thanks so much for the early encouragement, it made learning an enjoyable adventure instead of a tiresome task. Many thanks to Major Phil Amburn, my advisor. Without his patience and understanding this would never have been completed. To the other students in the graphics program who helped me with those minor setbacks which quickly became more than a molehill of a problem to solve. My gratitude also goes to Armstrong Aeromedical Research Laboratory for sponsorship of this effort.

Most of all, special thanks to my wife and family. The days through the program were demanding and you helped me make it through. You sat and listened with understanding, patience, and encouragement to help me carry on. I really did need all of your unending help, love, and devotion to complete this thesis.

Gary Kim Lorimor

## *Table of Contents*

	Page
Preface . . . . .	ii
Table of Contents . . . . .	iii
List of Figures . . . . .	vi
List of Tables . . . . .	vii
Abstract . . . . .	viii
I. Introduction . . . . .	1
1.1 Background . . . . .	1
1.2 Thesis Statement . . . . .	3
1.3 Scope . . . . .	3
1.4 Assumptions . . . . .	4
1.5 General Approach . . . . .	4
1.6 Presentation . . . . .	5
II. Overview . . . . .	6
2.1 Fast display of three-dimensional objects . . . . .	6
2.1.1 Shading Algorithms . . . . .	7
2.1.2 Adaptive Refinement . . . . .	8
2.2 Database Selection . . . . .	9
2.3 Red Flag Data Representation . . . . .	10
2.3.1 Header Record . . . . .	10
2.3.2 Static Data Record . . . . .	10
2.4 Dynamic Data Record . . . . .	12

	Page
2.4.1 Filter Output Data Block . . . . .	12
2.5 Polhemus Tracker Software . . . . .	16
2.5.1 Current Hemisphere . . . . .	17
2.5.2 Polhemus Output List . . . . .	17
2.6 Summary . . . . .	17
III. System Requirements . . . . .	19
3.1 General User Requirements . . . . .	19
3.2 Specific Requirements . . . . .	20
3.2.1 User Interface . . . . .	20
3.2.2 Hardware . . . . .	21
3.2.3 Software . . . . .	22
IV. System Theory and Design . . . . .	23
4.1 General Considerations . . . . .	23
4.2 Design Considerations . . . . .	24
4.3 Data Representation . . . . .	25
4.3.1 Data Filter . . . . .	26
4.3.2 Data Format . . . . .	26
4.4 Model Generation . . . . .	27
4.5 Key Frame Animation . . . . .	28
4.6 Summary . . . . .	29
V. System Implementation . . . . .	30
5.1 Filter Program . . . . .	30
5.2 Main Module . . . . .	31
5.2.1 Polhemus polling, Polhemus position report . . . . .	32
5.2.2 Time of data record . . . . .	32
5.2.3 Object Movement . . . . .	33

	Page
5.2.4 User Input . . . . .	33
5.2.5 Determining the viewpoint . . . . .	34
5.3 Polhemus Unit . . . . .	34
5.3.1 Initialization . . . . .	34
5.3.2 Polling the Polhemus Tracker . . . . .	35
5.3.3 Retrieving Polhemus Tracker Information . . . . .	35
5.4 Data Handling . . . . .	35
5.4.1 Traversing the Data . . . . .	35
5.4.2 Move to a Specified Time . . . . .	36
5.5 Display Unit . . . . .	36
5.5.1 Initialize Objects . . . . .	36
5.5.2 Object Movement . . . . .	36
5.5.3 Object Display . . . . .	37
5.6 Summary . . . . .	37
VI. Conclusions and Recommendations . . . . .	38
6.1 Conclusions . . . . .	38
6.2 Recommendations . . . . .	39
Appendix A. Calculation of the Viewing Parameters . . . . .	40
Appendix B. Guide for Filtering Data . . . . .	48
Appendix C. User's Manual . . . . .	49
Bibliography . . . . .	51
Vita . . . . .	52

### *List of Figures*

Figure	Page
1. Euler Angles . . . . .	15
2. X Direction Cosines . . . . .	16
3. Wire Frame Model . . . . .	28
4. Determination of Line of Sight . . . . .	40
5. Translation to Eye Coordinate System . . . . .	41
6. Ninety Degree Rotation about the X Axis . . . . .	42
7. Rotation about the Y' Axis . . . . .	43
8. Rotation about the X' Axis . . . . .	44
9. Reversal of the Z' Axis . . . . .	45
10. Rotation about the Z' Axis . . . . .	46



*List of Tables*

Table	Page
1. High-Activity Data Block . . . . .	11
2. Filter Output Data Block . . . . .	13
3. Comparison of Workstations . . . . .	25

*Abstract*

The purpose of this investigation was the development of a software system to integrate time-dependent data with a three-dimensional virtual environment. Red Flag data tapes were used as a source of time-dependent data. To project this into a virtual environment a head-mounted display was used. The software development was done on a Silicon Graphics Iris 3130 workstation. The design of the software system allows the user to be free of the keyboard for most situations. The position of the user's head was determined using a Polhemus 3-space tracker. The user could exercise basic control of the system with input from mouse buttons. The user was allowed to move to any position in the environment by looking in the direction of desired travel. The user was also given the option to ride in any cockpit which was in the environment. This allowed the user to see the environment from a pilots perspective at a specific point and time in the display sequence. Minimal machine dependent routines were used in the development of the software system. The software system was developed using the C programming language. The results of this effort were encouraging. There appears to be further possibilities of use for a system of this type in the training arena. The update rate of this system is about 10 frames per second. There appears to be several open questions concerning the benefits of such a system in the training environment.

# REAL-TIME DISPLAY OF TIME DEPENDENT DATA USING A HEAD-MOUNTED DISPLAY

## *I. Introduction*

This thesis deals with the development and integration of a software system which allows a person to view a time-dependent data set within a virtual world. The virtual world is brought about by the use of a head-mounted display. The time-dependent data base for this particular system is extracted from Red Flag exercise data tapes.

### *1.1 Background*

The idea of using a head-mounted display to project a virtual world to a person is not a new one. This idea was first proposed by Ivan Sutherland in 1965. He envisioned the use of a head-mounted display to project a virtual world to the user and put the user directly into the environment. However, the technology needed to achieve this was not available at the time (3:9).

With the passage of time there has been an evolution in the field of electronics. The computer has changed from a large, costly machine to operate to a laptop size which is affordable to the common man. The execution time of the instruction set for computers has decreased significantly. The television has evolved much the same way. We have gone from large black and white televisions which occupy the corner of a room, to color sets which can be held in the hand. This evolution of technology now allows the manufacture of a true head-mounted display. The head-mounted display can now be built at a reasonable cost with color displays and unlimited movement by the user. This allows a person to view a virtual world (environment) in a manner which is second nature to him. If a person wants to see what is occurring around him, then he merely looks at the area he is interested in. This is the same manner they have observed things for years. By providing depth cues such as size difference or stereopsis the environment can be made even more believable. In this environment we can project anything we want from molecules to mountains or any other

desired article. Then allow the user to move around within this environment as desired. Thus giving the user the opportunity to find the answer they are seeking. This answer may not be as apparent to them from a regular display or stack of numbers.

This technology can also be applied to another environment which has advanced a great deal in complexity with the advances in electronics, the cockpit of a modern day fighter. With advances in the complexity of fighter aircraft and weapons systems, the pilot's job becomes increasingly difficult. He must manage the weapons system to counter any threat while still accomplishing the mission. This is hard enough in the training environment let alone the combat environment. To ease the burden of the pilot in such a hostile environment, a means to present information to the pilot in a more efficient, usable manner is necessary. Situational awareness and effectiveness can be increased when information is presented in, three dimensional, graphical images instead of presenting numerical data (5:86). By giving the pilot the information in a manner which needs little translation or interpretation the task becomes much easier to accomplish.

By displaying the data in the pilot's field of view, the need for the pilot to remove his attention from the combat environment is reduced. This reduces the workload placed on the pilot by the complex and demanding environment of modern aerial warfare.

The use of a head-mounted display would allow the projection of information into the pilot's field of view while he is looking outside the cockpit. The combat environment is not the only place where this type of information display may prove useful. A head-mounted display may be useful in a training environment also. This could be useful to debrief pilot's after a practice sortie. The data which has been stored from a practice engagement could be replayed through a head-mounted display. This would allow the instructor and pilot to get a view from another person's perspective. Seeing the movement of aircraft from the same viewpoint as the person who was there only a short time before. To this extent the viability of a head-mounted display is studied to see if a head-mounted display would truly be feasible to project a virtual environment to a pilot or other person and to see if there is a possibility a head-mounted display could assist in the enhancement of the training at Red Flag training, other exercises, or the combat environment.

## *1.2 Thesis Statement*

A computer software system using a head-mounted display device can be designed and implemented using graphics-based software to allow the user to effectively view and understand a time dependent environment. The environment used in the application would come from an air tasking order (ATO) or telemetry data from exercises such as Red Flag, Green Flag, or other training exercise. Further, the application must display quality, time dependent, three dimensional images in real-time or near real-time. The overall goal of this application was to allow the user to view an environment from an advantageous viewpoint (like a bird's eye view) and allow the user to move about within the environment. The user is allowed to occupy any object within the environment and move with that object. This object could be an airplane, tank, car or any other vehicle which is in the environment. This gave the user the added benefit of seeing the environment from any of the participant's viewpoints.

## *1.3 Scope*

This research effort dealt with the development of display algorithms for high quality, time dependent, three dimensional images displayed in real-time or near real-time; development of software routines to filter telemetry data from numerous Red Flag exercises; and additional software to allow the interface of a head-mounted display equipped with a Polhemus tracker to allow a more natural, easier user-machine interface. Near real-time will be defined as 12 frames per second for this application (10).

The software application was targeted for a Silicon Graphics IRIS 3130 Workstation. Software was written to ease transporting the software to another system. Development of the application software required investigation into six areas:

1. Methods proposed and/or used to generate high speed three-dimensional graphics images.
2. Methods proposed and/or used to generate high quality shaded images in real-time or near real-time.
3. The selection of a suitable database to represent the situation model.
4. Conversion of the Red Flag data tapes into a format usable by the graphics software.

5. Software routines to allow the efficient use of a Polhemus tracking device.
6. Integration of Kalman filtering techniques to aid in the predictive tracking of head movements.

#### *1.4 Assumptions*

The user of this system could be one of several people. A cockpit display researcher interested in interactively working with the engagement environment and researching the aid of this application to a pilot. Another user could be a person on a battle staff viewing a previously planned exercise in simulated operation. A person could also use this for tactical mission planning to view the planned mission ahead of time. An additional user would be an instructor reviewing and critiquing a recent exercise from data tapes generated from telemetry data. Another possibility is a student seeing a situation for the first time before attempting to try the maneuver on their own.

Working within the environment consisted of moving from an above-environment view to any participant's view within the environment. The additional capability to change to a specified time during the engagement was included. This system is an open loop system. The user can only control the viewing time or what they see. They are not able to control the actions taken by any of the objects within the engagement. To this extent the user is limited to a choice of display rates and time of display; no parameters about actual object movement can be changed. The user can change the viewpoint at any time by moving within the environment. He can also change what he sees at any time by looking somewhere else.

The target system that was to host this application used the C programming language. Hence, all the software development and implementation was done in C.

#### *1.5 General Approach*

After the five subject areas were adequately researched, software development began using a prototyping design approach. Software prototyping involved a modular oriented design approach. With modular construction, the software was easier to modify. All software calls to the graphics library routine were isolated as much as possible to minimize

system dependence. This type of software construction would allow an easy transfer of the software to another computer graphics workstation. Software development would be complete when:

1. Three dimensional images were displayed on the head mounted display in near real-time.
2. The engagement data base could be manipulated to specific point in time of the engagement sequence.
3. Flat shaded images were generated and displayed from the engagement database.
4. The user could move anywhere desired in the environment.
5. The user could travel with any object within the environment.

#### *1.6 Presentation*

This thesis is made up of six chapters and an appendix. Chapter II presents a survey of the literature in each of the areas mentioned in the Scope. Chapter III covers a description of the software. Chapter IV discusses the system theory and design, while chapter V covers the system implementation. Conclusions and recommendations are covered in chapter VI. Appendix A describes the development of parameters necessary to view the scene from the users viewpoint. Appendix B describes the use of the software tools developed. The user's manual for the software system is found in Appendix C.

## *II. Overview*

There were five areas of interest which had to be researched before any efforts in software design could begin. These areas were quick display of three-dimensional objects, quick methods of generating shaded images, database selection to represent the situation model, conversion of the Red Flag data tapes to a suitable format, and software routines to use the Polhemus tracker. These areas are reviewed in the following sections.

### *2.1 Fast display of three-dimensional objects*

There are several steps which must be taken to properly display a three dimensional image. The object must be transformed from world coordinates to viewing coordinates of the display system through several transformation matrices. This viewing pipeline of transformation matrices may be combined as much as possible to reduce the number of multiplications involved. The object is first transformed from the world coordinate system where it is described to viewing coordinates. From there the model is transformed from viewing coordinates to device coordinates through a sequence of steps which clip the object to the viewing volume, project the object onto the viewing plane, and transform the object into device coordinates (6:284). Each one of these steps takes a finite amount of time. The total time through a three dimensional viewing pipeline can become quite severe when when additional operations such as shading are performed upon the image. With the requirement for shading in real-time the difficulty of real-time display is greatly increased. One possible way to increase the speed of display generation is by preprocessing the data into a binary space partition (BSP) tree (8:65).

A BSP tree divides the data set into an efficient structure for the quick rendering of a perspective view. Another way to speed up the display of a shaded image is do as much processing of data before display time as possible. This is also know as preprocessing of data.

To give an idea of the complexity required to render a nicely shaded image, look at the efforts required to develop an image into a usable form. The authors (8) had their own comment about the rendering of images:



It is not surprising that the generation of rendered color images is computationally expensive. Not only do the usual transformation, clipping, and perspective steps need to be performed but visibility and rendering calculations must also be performed for every pixel in the image (8:65).

To shorten this processing time and get near real-time three dimensional images the visible-surface overhead must be put into a preprocessing phase (8:65). This is only possible if the image of the world changes infrequently. To get the needed speed of image generation a BSP tree is used (8:66). With a static world model and sufficient local memory a BSP tree is easier to manipulate than other three dimensional rendering algorithms (8:68). The algorithm consists of two components: 1. Conversion of the input polygon list into a BSP tree structure prior to run time. 2. Generation of polygons in back to front order by traversal of the BSP tree structure by an image generation module (8:67). This allows quick three dimensional image construction by placing the motionless objects into a tree structure. Now, the objects are drawn from back to front based on the location of the viewpoint. The hidden surface problem is solved in this manner with the need for many of the time consuming calculations eliminated. However, no provisions are made for shading of the image. This would be accomplished by another routine in the display pipeline.

*2.1.1 Shading Algorithms* Phong shading calculates the difference between the surface normal at a point and the vector halfway between the eye and light vector. This specular light causes highlights on surfaces due to reflected light (7:577),(11:401). Other terms included in the calculation of Phong shading are the ambient light; light from reflections off of every surface in the image and diffuse light; light from the light sources shining directly on an object. Phong shading is not used in real-time graphics due to the time required for shading computations (2:103). Phong shading can be made more efficient by rearranging the equations which model the light (2:104). By rearranging the equations for the calculation of light values the equations can be reduced considerably. Three additions, a division and a square root calculation are still required. Division and square root calculations are too slow for real-time applications. To improve the performance use a Taylor series approximation for division calculations. This will remove the need to divide. For the square root use a table look-up to find the result (2:104). These modifications will re-

duce the computational overhead to five additions and one memory access/pixel. By using these calculations Phong shading may be improved to be useful in real-time applications. Another method is to use flat shading of the images. Flat shading is based strictly on the angle of incident light to the surface normal (6:576).

The choice of shading model is dependent on the degree of realism required by the user. If the need is for a believable scene, yet not a totally realistic scene, the geometry engine of the Iris could be used and greatly shorten the display time. The Iris has a built in VLSI geometry engine to handle many fundamental graphics operations (14:20). By combining the efficiency of the available hardware with the advanced algorithms it may be possible to display three-dimensional shaded objects in real-time. If this method doesn't give the update rate required, another method is to show objects as dots or wire frame figures and refine them selectively when there is no more motion.

*2.1.2 Adaptive Refinement* Another approach to assist in the necessary response time is to give as much of the information as soon as possible and improve the image as time progresses (1:29). This technique has been effectively used to display an image of increasing quality depending on the time the user remains looking at the image. "The idea is to store a hierarchy of object descriptions and choose the most appropriate representation at display time" (1:30).

When displaying an image using this technique the first image would simply show a pattern of dots representing the vertices of the image. As soon as possible after this polygons (the interconnection of an ordered list of the vertices (11:231)) would be shown as a wire frame diagram displaying the image.

Following this the display would be updated by creating an image using a flat shading algorithm. "Flat shading calculates a single intensity value for shading an entire polygon (6:580)." This would give a three dimensional representation of the image with the surfaces having a faceted appearance. The next operation performed on the image would improve the image to give a smooth shaded image by using a different shading model. This model could either be generated with Gouraud shading or Phong shading. The option also exists for using both shading models.

The last item accomplished in this display pipeline would be to anti-alias the display. This is also called dejagging of the image (7:436). The operation reduces the stair step appearance of a diagonal line crossing a limited resolution display. Anti-aliasing may be very time consuming depending on the algorithm used.

The advantages to the adaptive refinement type of algorithm would be a quick display of the image with reduced quality. This would allow a believable image to be generated in real time. In this application the user would get increased detail by stopping the cycle of events and keeping their head in a fixed position for a predetermined amount of time. The longer a user kept the head in a fixed position the more detail which would be shown. "This allows the user to decide when the image is detailed enough" (1:30). The amount of data manipulation could be reduced by only updating the necessary data as quality improves.

There is a disadvantage when using adaptive refinement. The algorithm may not allow the early images to be useful due to their low quality (1:31). Once the display pipeline has been traversed and the rendering of the image is complete there is no further improvement in the quality of the image (1:31). With the above display techniques a speed update rate may be possible to allow the generation of real-time high quality images.

## *2.2 Database Selection*

The selection of the structure of the database is really dependent on the type of manipulations which will be accomplished. By using a data structure which is matched to the operations on the data the efficiency of the program will increase. The data used in this software system will not be modified by the system at all. The principle operation will be to traverse the data in time order. That is, the events should be traversed in the order which they occurred. The logical choice for a structure would be a list. In this situation the best list to use appears to be a structure of arrays. If this list is ordered in an array and each element of the array is a C structure containing the necessary information, then the data could be quickly and easily accessed. If the user decides to go to another time in the sequence then the pointer to the current event would only need to be adjusted to the new event time. This is a logical layout for the database structure. However the structure

within the array elements still needs to be decided. This structure is dependent upon the necessary information which is required to display the objects.

### *2.3 Red Flag Data Representation*

To properly display the aircraft used in Red Flag training missions only a few parameters about the aircraft need to be known.

1. Is the aircraft a member of the red or blue forces?
2. What type of aircraft is it?
3. What is the aircraft's position?
4. What is the aircraft's orientation?
5. What time is this event occurring?

In this prototype, these are the questions which will be answered. Later implementations may want to include threat or weapons information.

This is all the information necessary to display the aircraft in this implementation and show the event time. The information available and the information required will be discussed first. This will be followed by a discussion of the structure of the elements in the database. The data from Red Flag contains vast amounts of information. This data is divided up into several sections (4). The major sections are the header record, the static data record, the dynamic data record, and the watchdog record. The function of each of these records is discussed below.

*2.3.1 Header Record* The header record provides information to label each tape. The header record contains the date of the exercise, reel number, and the site location (4:3-308).

*2.3.2 Static Data Record* The static data record provides information on activities in the exercise which change infrequently. This information includes high-activity participant data, sun and atmosphere parameters, remote site positions, threat mission data,

range status data, low-activity participant data, and other input parameters (4:3-309 - 3-310). Based on the specifications above the only area to find the data is the high and low activity participant data. By using only high-activity participant data the process will be simplified even more. The disadvantage of using just high-activity data is the interactions of the high-activity aircraft data with the low-activity aircraft is not shown. However, the intent of this project is a proof of concept prototype not a full fledged implementation, so only the high-activity aircraft will be shown.

**2.3.2.1 High-Activity Data** The high-activity participant data message contains several components as illustrated in Table 1. These components contain a great deal

Word	8 Bits	8 Bits	8 Bits	8 Bits
0	Message Label	Blocks	Words	
1	Mission Id	Color	A/C Number	A/C Type
2	Pod ID		EW Designator	AII Designator
3	Aircraft Tail Number			
:				
4				
5	Pilot Squadron or Call Sign			
:				
6				
7	Pilot Name			
:				
9				
10	G Limit	AOA Limit	IAS Limit	
11	Spare		Descent Limit	
12	WPN Type(1)		WPN Type(2)	
13	WPN Type(3)		WPN Type(4)	
14	WPN Load(1)	WPN Load(2)	WPN Load(3)	WPN load(4)
15	Repeat Words 1 to 14 for up to 36 Aircraft			
:				
504				
505	Vertical Parity Word			

Table 1. High-Activity Data Block

of unnecessary information. The information which is usable from this data is the aircraft type, color, and slot position. The aircraft type can be in the range from 0 to 84. This range of numbers relates directly to the make of aircraft and the telemetry pod position

on the aircraft. The color identifies the aircraft as a member of the red or blue forces. The slot position is used to identify which slot the aircraft information is stored in(4:A-25). These slots number from 1-36. The slots are used instead of the aircraft call sign or tail number to relate aircraft information with aircraft type. This eases the task of keeping track of high-activity participants.

Only some of the necessary information is available in the static data area. The dynamic data area must be examined to find the remaining necessary information.

#### *2.4 Dynamic Data Record*

The dynamic data area contains the information which is constantly changing. This data varies from communications between sites to Federal Aviation Administration input data (4:3-321 - 3-322). Much of this information doesn't even concern the position of aircraft. The block of data which contains the necessary information, about aircraft position and orientation, is the filter output data block.

*2.4.1 Filter Output Data Block* The amount of data available in the filter output data block can be seen in Table 2. Even in this block there is a significant amount of information which is unnecessary and will be ignored.

Word	8 Bits		8 Bits		8 Bits	8 Bits
0	Block Type 314				Block Length 49	
1	Hours		Minutes		Seconds	1/100 Seconds
2	1/1000 Seconds		Slot (1 to 36)		Pod ID	
3	Range Positions X, Y, Z					
:						
5						
6	Velocity Components X, Y, Z					
:						
8						
9	Acceleration Components					
:						
11						
12	Angular Rate Components					
:						
14						
15	Orientation Roll					
16	Orientation Pitch					
17	Orientation Heading					
18	Indicated Air Speed					
19	True Air Speed					
20	Mach					
21	Angle of Attack					
22	Angle of Sideslip					
23	Normal Acceleration					
24	Direction Cosine Matrix					
:						
32						
33	U-Bit	P-Bit	Spare	Group	Itrace	
34	Master ID				Interrogator ID	
35	Spare					
:						
38						
39	Inertial Angle of Attack					
40	Inertial Angle of Sideslip					
41	Corrected Loop Ranges 1 to 7					
:						
47						
48	Barometric Height of Aircraft in Feet					

Table 2. Filter Output Data Block

This block contains the necessary position and orientation information on each high-activity aircraft. The aircraft in this data block are identified by the slot numbers set up in the static data block. In this block are the time, range position, velocity, acceleration, orientation, air speed, direction cosine matrix, and other information relating to the aircrafts flight parameters (4:3-324 - 3-325). These parameters are updated about every 400 milliseconds for each aircraft. This gives five updates for every six seconds, roughly three updates per second. The time, position, and orientation are still needed to specify a complete event.

*2.4.1.1 Time* The time stamp available for each event is accurate to the  $1/1000^{\text{th}}$  of a second. However this type of accuracy is not required. Only the hours, minutes, and seconds will be required. Event data will be extracted once for every second. Using data once for every second allows recovery of data in the event of a data error. This will give a good chance of having reliable position and orientation data for each aircraft, for each second. The values for the time are in integer format and present little problem in reading.

*2.4.1.2 Aircraft Position* Aircraft position is based on an X, Y, Z reference system. This reference system is the normal right-handed coordinate system with the origin defined as  $37^{\circ}$  N latitude and  $115^{\circ} 30'$  W longitude. From this origin the Northerly direction is in the positive Y direction, Easterly is in the positive X direction, and positive Z is the direction of increasing altitude (4:3-336,3-338). The X, Y, and Z values for aircraft position are directly available from the filter output data block. The numbers are in the Perkin-Elmer floating point format and need to be converted to a floating point format compatible with the machine being used. Once the position of the aircraft is established, the orientation of the aircraft needs to be established for proper display.

*2.4.1.3 Aircraft Orientation* The use of orientation is necessary to accurately present the aircraft's attitude on a display. The orientation gives the rotations of an object in relation to the reference frame being used. In this case the reference frame is the fixed reference coordinate system described above. There are several means to describe an



aircraft's orientation. The most common method is the Euler angle sequence of azimuth, elevation, and roll as illustrated in Figure 1. This definition has the advantage of using only

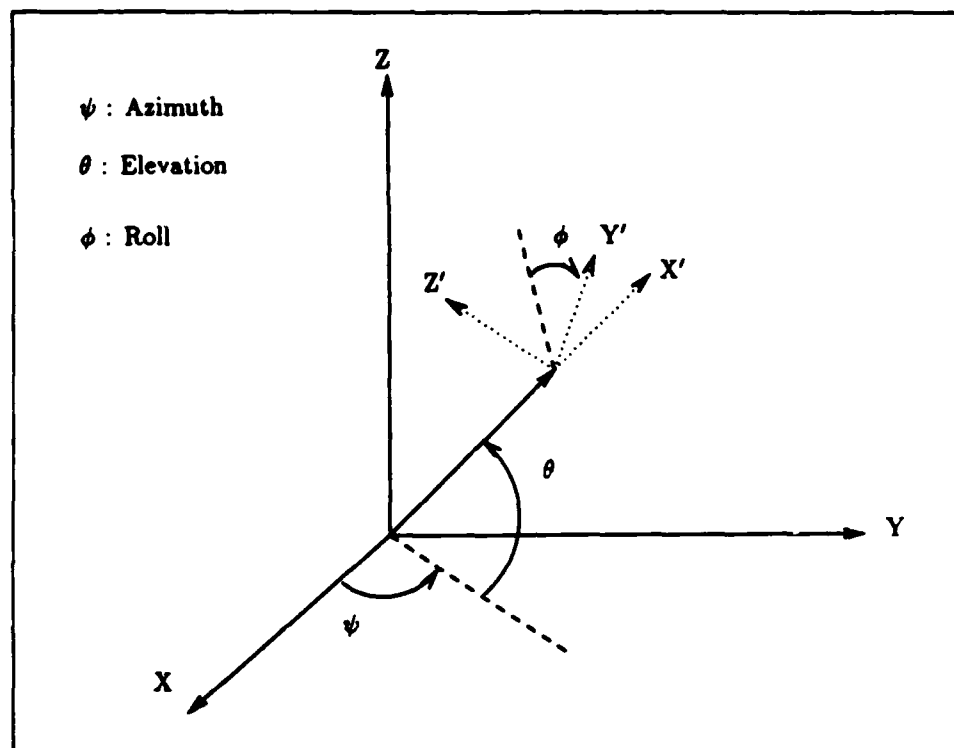


Figure 1. Euler Angles

three numbers to specify the orientation, so the form is compact and efficient. However to convert these angles into a form usable in the transformation matrix, i.e. to convert from the reference frame to the display coordinates, several floating point calculations must be accomplished.

Another method to describe the aircraft orientation is with direction cosines. Direction cosines are the cosines of the angles between the aircraft's X, Y, Z axes and the X, Y, Z axes of the measurement reference frame (12:B-3). So the X direction cosines would consist of the cosines of the angles between the aircraft's X axis and the reference X axis; the aircraft's X axis and the reference Y axis; and the aircraft's X axis and the reference Z axis. This is illustrated in Figure 2. This form is not as compact as the Euler angle sequence but, it is much more efficient. The direction cosine matrix can be inserted into

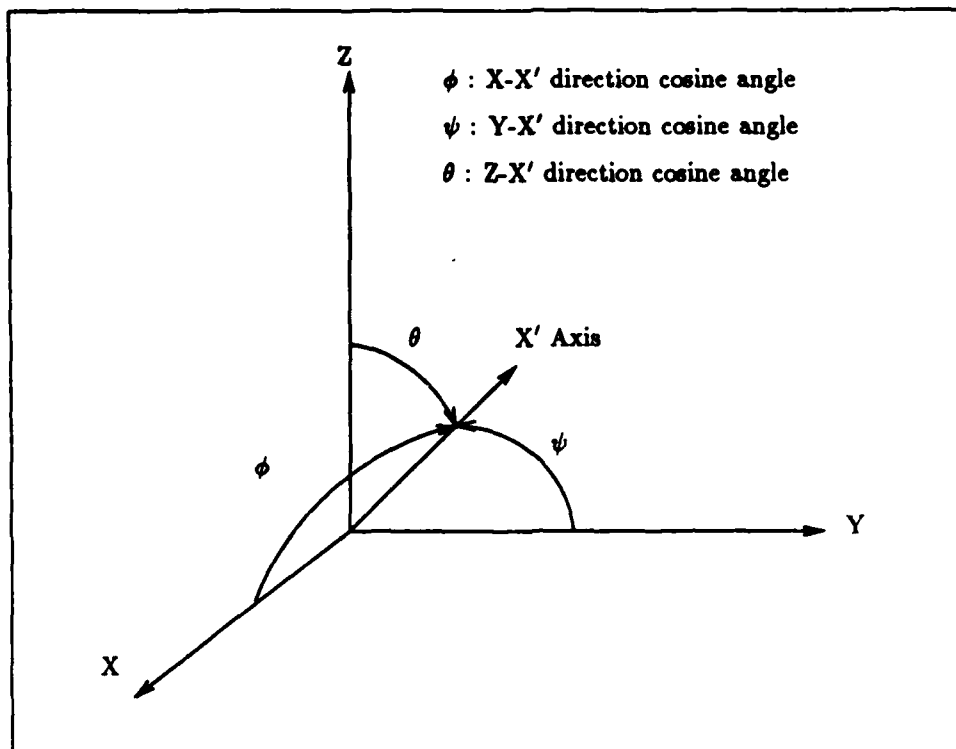


Figure 2. X Direction Cosines

the transformation matrix directly with no floating point calculations necessary. This is possible since the direction cosines are mutually orthogonal unit vectors (6:258). When the direction cosines are combined with the position of the aircraft, a transformation matrix can be constructed which accurately describes the relationship of the aircraft to the reference frame.

### 2.5 Polhemus Tracker Software

To properly display a mission, a device was needed to constantly monitor the orientation of the user's head in three-dimensional space. A device with the needed characteristics is a Polhemus tracker. The Polhemus tracker uses low-frequency, magnetic field technology to determine the position and orientation of a sensor in relation to a source or other specified reference frame. This information may be transmitted to a host in ASCII or binary format (12:1-1). To properly use the Polhemus tracker the device needs to be initialized

to obtain the proper data. This requires the definition of the current hemisphere and the output list of the Polhemus tracker.

**2.5.1 Current Hemisphere** Because of the symmetry of the magnetic fields generated by the source, there are two mathematical solutions to each set of sensor data processed. Therefore, only half of the sphere surrounding the source is practically used at any one time without incurring any ambiguity. This half sphere is referred to as the current hemisphere. The chosen hemisphere is defined by an line-of-sight vector from the source through a point at the zenith of the hemisphere (12:5-31). Once the current hemisphere is properly specified, the Polhemus tracker needs to be initialized to return the proper output list.

**2.5.2 Polhemus Output List** The output list refers to the data items to be included in the data record output by the Polhemus tracker (12:5-44). This list has available several options including direction cosines, quaternions, and orientation coordinates (12:Appendix F). For this application only the orientation angles will be necessary to determine where the user is looking.

## **2.6 Summary**

Several possibilities were considered during the design and implementation of this software system. There was hope of solving the hidden surface problem, with all the objects in the scene, in real-time. This may have been possible if all the objects were static or did not move frequently. Then a BSP tree could be used to efficiently partition the model for rapid display. However, with dynamic objects in a scene this is not possible. The time involved in restructuring the BSP tree for each image would not provide the required image update rate. An intermediate solution is to use a BSP type algorithm with the static data only. Then once the static data is displayed, display the moving objects. This does not allow the hidden surface problem to be solved for the moving objects. However the display can be accomplished in real-time. The shortage of processing time also occurs when trying to shade images in real-time. To provide a shaded image with the hidden surface problem solved three options were considered. Either increase processing speed, reduce the image

update rate, or accept an image which does not solve the hidden surface problem with shaded images. Adaptive refinement would be a good idea to implement and may prove useful. However, with the initial prototype the usefulness of displaying time dependent data with a head-mounted display should be established before adaptive refinement and solutions to the hidden surface problem are solved.

### *III. System Requirements*

With the development of a three-dimensional virtual world system (head-mounted display) at AFIT, there was a question of how to use this new display device. After all, any virtual world can be displayed on such a device. Examples of such worlds are the Indianapolis 500, Americas Cup Race, or a Red Flag exercise. The subject decided upon, was a world which relates directly to the Air Force, Red Flag. Red Flag is a training environment for air-to-air combat. At Red Flag pilots fly training missions to improve their flying skills in a simulated combat environment. These missions are recorded on data tapes which are used for review after the mission to improve the training value of each mission.

This chapter details the user requirements for this thesis effort. At the beginning of this thesis effort very few requirements were specified. As the thesis progressed, the requirements developed into a need to provide software with a user interface which was easy to use. This chapter will discuss the general requirements and capabilities of the system. This will be followed by a discussion of specific requirements concerning the user interface, computer hardware, and software.

#### *3.1 General User Requirements*

Overall the requirement was to provide the user an environment which would allow the ability to review a time dependent situation in real-time or near real-time. For this thesis, the time dependent situation would be actual data extracted from Red Flag data tapes. To this requirement the capability to view the situation from a bird's eye view, or from any object within the situation was added. From these two requirements several other requirements were derived.

One of these additional requirements was to modify the virtual world in response to the viewer's motions. To accomplish this a Polhemus tracker would provide orientation information concerning the user's orientation in relation to the virtual world. This had to be done in a reliable manner with little delay from the receipt of information from the Polhemus tracker to the change of the displayed view.

Another requirement was to provide realistic motion to the aircraft from the Red Flag data and put this data into a format which is transportable between computers and easily understood. In addition the database extracted from the Red Flag data tapes must occupy less disk space than the Red Flag data tapes. The new database must be organized to allow quick traversal to assist in the display of the situation in real-time.

The real-time display of models based on the information contained in the extracted database was the next requirement. To realize this requirement simple yet identifiable models constructed of few polygons would need to be designed and constructed.

The last requirement was to give the user as much freedom as possible in the virtual world. The implementation of a user interface to the system which would allow the user to remain in the environment without the use of a keyboard for input was necessary. The user must be separated from the computer to give the freedom of motion necessary to move around in the virtual environment created by the head-mounted display. From these general requirements, several additional requirements fell out which were more specific.

### *3.2 Specific Requirements*

Several additional specific requirements were needed to clearly define the manner in which the software would be developed and the manner in which the user interface would operate. The user interface to a virtual world environment is different than the interface to most environments. With the common workstation the user sits in front of a CRT and controls the display of data through keyboard input, mouse input, or the use of some other input device. This world is stationary unless there is some type of input to change the display. The head mounted display, on the other hand, projects the virtual world right before your eyes and you could not see a keyboard if one were available. This world does not remain stationary, instead this virtual world follows your head movement and the proper image is projected for the orientation of your head. A different means of input to the system, other than the keyboard, is required for good user interaction.

*3.2.1 User Interface* The user had to be able to freely move around in the environment created by the head-mounted display. The user was given the option of being able to

view the database at three different speeds. The speed for viewing the database would be normal speed, half speed, and double speed. The normal speed would display the database at the speed in which the actual mission occurred. Half speed would display the database in twice the actual time of occurrence. Double speed would display the database in one half the actual time of occurrence. This would allow the user to watch in detail those interesting parts of the situation which passed too quickly and allow the user to quickly view any part of the situation which was not as interesting.

Additionally the user was given the capability to proceed to any desired time within the situation. This allows the user to review any part of the situation as many times as desired.

For the user to freely move around the virtual world, the user had to be separated from the keyboard. To separate the user from the keyboard, the input of basic commands from a mouse or other input device was necessary to allow the user to control the system. The commands available to the user with a mouse are limited and would only allow limited interaction with the system. Keyboard commands allow the user to control the speed of the display sequence, to step to a specified time, or allow the user to select a specific aircraft to view the environment from. The capability for the user to exit the program from the keyboard should also be included.

**3.2.2 Hardware** The software system developed was required to display the images at a minimum of 12 frames per second. These images contained wire frame figures which were constantly in motion. The hardware to display these images needed the processing speed and graphics capability to give the required display update rate. This quickly limited the selection of computers with the required capability. One, the Silicon Graphics Iris 3130 Workstation (Iris), was completely capable and would definitely be used in the development of the software system. Testing was conducted on the Digital Equipment GPX Workstation (GPX) to see if the software system would meet the requirements while running on the GPX.

**3.2.3 Software** Software implementation required the use of good coding practices be followed. The code should be as modular as possible with many comments to allow for easy modification. In addition the machine-dependent code would be isolated as much as possible and kept to a minimum to allow ease of transportability between different computer systems.



#### *IV. System Theory and Design*

This chapter covers the theory and rationale of design decisions made during the development of the software system. General design considerations are discussed first. Following this a short description of other considerations are presented. Then data representation issues are covered. Model generation is discussed next. Then a discussion of the smoothness of display via key-frame animation is discussed. Finally, a summary will end the chapter.

##### *4.1 General Considerations*

The general design of the software needs to be as simple as possible. This simplicity should be designed into the construction and into the ease of using the software for this application. Simplicity in construction will assist in easy modification of software to extend the capabilities of the software in the future. Simplicity of design in the user interface will allow ease of use and encourage the application to be used.

The aircraft in the database needed to be represented by some means. There were several possibilities to consider. Icons, wire frame models, and shaded images were considered. The best choice appeared to be a filled image when viewing the aircraft. When the user was in the cockpit this caused a problem since a great deal of the environment was blocked by the filled in image. To allow the user to view the situation around them when in an aircraft cockpit, a wire frame was used. This allowed the user to look around with unrestricted view and still provide references about the aircrafts orientation.

The data base used to position the aircraft could be stored on disk or in memory during program execution. Memory was chosen over disk storage due to the difference in access time from disk vs access time from memory (9:327). By keeping the disk access to a minimum during operation, the display update rate was increased. The use of virtual memory could cause a problem if the algorithm for paging was inefficient (13:84), but this was not a factor in this software system.

To minimize the delay between the retrieval of necessary data and the display of the data, the inner loop must be as small as possible. The inner loop needs to perform the following operations as a minimum:

```
loop
    poll the Polhemus tracker
    get the next data position
    check for user input
    retrieve Polhemus tracker orientation
    update the viewpoint
    display the image
end loop
```

By keeping the inner loop as small as possible more time could be spent updating the display, giving a higher display rate and smoother aircraft motion. To make this inner loop as short as possible several matters were considered in the design of the software system.

#### *4.2 Design Considerations*

While designing the software there were several items of interest. Human factors were investigated to help design an interface which was easy to understand and use. Workstation capabilities were considered to find the best workstation available for the intended application. Machine dependent library routines were minimized to allow as much transportability as possible.

A short study was done on the available workstations to determine which workstation would be the most suitable to meet the software requirements. The workstations used in this study were the Iris and a GPX. Both of these machines have a graphics capability and are designed to handle high display rates. Their other attributes are listed in Table 3

To evaluate these machines a program was written for a GPX which was already present on the Iris. This program was designed to display a wire frame model constructed from a set of points. Unfortunately the capabilities of the GPX in the graphics arena were

Colors	Iris	GPX
Display	4096	256
Processing Speed(MIPS)	Double Buffer	Single Buffer
Geometry Engine	≈1	≈3
	Yes	No

Table 3. Comparison of Workstations

limited. The result of the study found the GPX could process numbers much quicker than the Iris, but the GPX was much slower at displaying graphics than the Iris. Since speed of display was the most important factor, the Iris was chosen for software development.

Transportability of the software was also desired. When moving software from one workstation to another workstation one must consider the type of computer specific routines used and eliminate as many as possible. For most of the software this is not a problem. Since the majority of the software is used for processing data and little for display of data. A problem does occur in the most important part of the software system, the display routines. The use of a software graphics standard, such as CORE or GKS would be helpful in overcoming workstation differences. However, the use of these systems would be too slow, due to processing overhead, to allow the required display rates. For this reason, display routines unique to the Iris were necessary to gain the display speed for the required update rate of the display. Unfortunately this will make moving the software to other workstations more difficult than hoped for.

#### 4.3 Data Representation

The discussion from chapter two clearly shows what information is necessary for the proper display of data. Based on the previous discussion, it would be inefficient in both time and storage space to use the data as supplied. A better method is required to store the data. By reducing the size of the data less time will be required to access each element of data, allowing a faster update rate.

The database is divided up into two segments. The first segment will contain the necessary initialization information. This specifies which aircraft, both type and color,

is assigned to which slot position. The second segment of the database contains the orientation and position information about each aircraft arranged by slot position. This significantly reduced the size of the data structure and allowed efficient access to the database.

*4.3.1 Data Filter* The data filter was necessary to pare down the vast amounts of data received. In order for the data filter to be useful several qualities were incorporated into the design. From the documentation (4) provided a data filter was designed which would do the following:

1. Provide a database to the software system in a usable format.
2. Provide a data format easily readable and understandable.
3. Retain only data which is necessary for the display sequence.
4. Be easily modifiable to extract other data features which may be necessary with future enhancements.
5. Recover reliably from data errors inherent in telemetry data.

These items are discussed in greater detail below.

*4.3.2 Data Format* In addition to the aircraft position and orientation some additional information to describe each aircraft was necessary. Each aircraft had to be related to one of the models available. These models are simple wire frame models and allow the user to quickly identify the aircraft type. In addition, each aircraft had to be identified as a member of the blue or red forces. The result was a data format in two parts. One part would describe the aircraft type and identify them as red or blue forces. The other part, which is the major portion of the database, contains the position and orientation information for each aircraft for each particular time. To access this information quickly in a time ordered fashion, the data is put sequentially into a data file. This gives an ordered list which is easy to access.

*4.3.2.1 Readability* Easy readability is achieved by storing the data in an ASCII format. This allows a user to read through a data file if necessary. The disadvantage of using this format is the extra space required to store the data. However this

format also insures transportability to almost any machine. These two factors outweigh the disadvantage of the increased storage requirements.

**4.3.2.2 Necessary Data** The extraneous data available on the received data tapes is significant. About 90% of the data is unnecessary and needs to be passed over. This can be accomplished by recognizing each data type present and ignoring this data if the data is not required for aircraft display.

**4.3.2.3 Modifiability** The software is commented to allow the user to follow the program flow and which data set is being extracted or ignored. Each data type is recognized by the filter. Only the necessary portions of data are extracted. The other portions are skipped over. To extract a new data type two tasks must be performed. First, the routine must be written to extract the new data type desired. Second, the routine which recognizes the data type must be modified to call the new extraction function and not skip over the data.

**4.3.2.4 Error Recovery** The manner in which the filter recovers from errors can greatly affect the data set extracted. If the filter exits the program on an error then a significant amount of data can be lost. If the filter ignores all data after an error the effect is the same. To recover from data errors the filter should compare each character with the available data and try to find data which will synchronize the filter into the proper extraction sequences. The last approach is used for this filter. The filter recovers as much data as possible with no operator intervention. This is a very desirable feature since the data files are quit long (100 Megabytes or greater) and take considerable time to filter. Once the data has been filtered a means of representing the data in a visual manner is necessary.

#### **4.4 Model Generation**

The models describing each aircraft were required to be simple and easily identifiable. To achieve simplicity, the characteristic features of each aircraft were identified and used to describe the aircraft model. This simplicity gave the added benefit of a reducing the the

number of polygons necessary to display each aircraft. This task was one of the easiest to accomplish. Three view drawings of various aircraft were received with the documentation of the Red Flag data. These drawings were quickly converted into a standard polygon description file used by the software system. An example of the wire frame models produced is illustrated below. After the models were converted into the standard polygon file format,

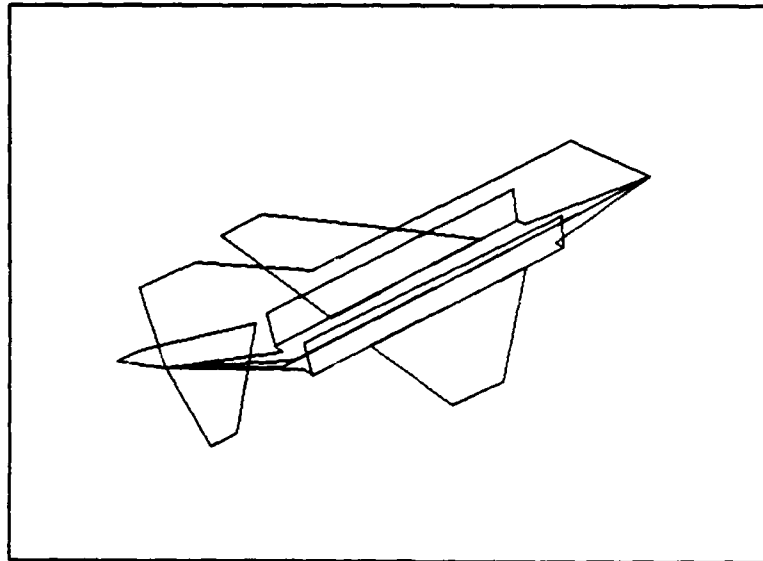


Figure 3. Wire Frame Model

the task of providing realistic motion was necessary. Key frame animation was used to accomplish this task.

#### *4.5 Key Frame Animation*

To get a believable animation each aircraft's position and orientation were required to be specified for each frame of the display sequence. There was insufficient data to show the aircraft position every frame of the display sequence. The other drawback to having the data available for every frame is the vast amounts of storage which are required to contain such a data set. To make up for the lack of data and the blank spots or poor animation which would occur if the picture were updated only when a new set of data were available for display, key frame animation is used. This process involves taking the present aircraft position and orientation and the next aircraft position and orientation and finding

the difference between these parameters. This difference is then divided by the number of frames to display between each new data point giving a delta for each value. As each frame is displayed this delta is added to each parameter to update the position and orientation for the next frame. When calculating the deltas a linear interpolation was used. Another possibility for calculating the deltas would be to use splines or some other higher order function to fit curves to the data points. However, this was not required for smoothness since the data was extracted from actual aircraft movement and the extracted data base contained sufficient data so that good linear interpolation provided realistic movement.

#### *4.6 Summary*

The data filter was a crucial component of the software system. The ability to show realistic movement without actual data would have been very difficult, if not impossible to generate. To create a believable environment, aircraft models were generated, displaying characteristic features of the aircraft being modeled. Key frame animation was used to give motion to the aircraft models and depict aircraft maneuvers in the Red Flag environment. With the data extracted, the models generated, and the animation method finalized, the software system was ready to be implemented on the Iris.

## *V. System Implementation*

This chapter discusses the process in which the software system was implemented. Each area of the software is discussed in detail. Initially, a filter program was implemented to convert the data into a usable form. Following this, the software system was implemented consisting of a main module and associated subunits. The main module has control of all the other subunits. The software is considered to be divided into three separate subunits. One subunit manages the requirements of the Polhemus tracker. Which is used for following head movement. Another subunit takes care of traversing the aircraft position data file. The last subunit controls the movement and display of the objects. The implementation of the filter program will be discussed first. This will be followed by discussion of the main module and the functions it carries out. Then a discussion of the Polhemus subunit, data subunit, and display subunit will follow.

### *5.1 Filter Program*

The filter program converts a set of data which has been read in using the "dd" program available on Unix based systems. This data is read into the program in 32 bit words. The words are then checked to see what type of data follows. Only static data records and the dynamic data records are used. The static data record contains much more information than needed (4:3-309 - 3-319). To identify the static data section the word "aircraft" followed by the number of high activity aircraft was used to identify the beginning of a static data area. This type of data changes very rarely and appears infrequently in a converted data set. Most often the static aircraft data is found in the beginning of the database file.

The other information needed was in the dynamic data. This data area specified significant information about each aircraft. The only information extracted was the position and orientation information. To make this easily accessible the information for each aircraft was saved in an array indexed by the aircraft's slot position. This information was stored until the beginning of the next time period. When the next time period arrived the data in the array was stored. In this adaptation one second was used to define the



next time period. When the next time period arrived the data was saved to the converted file and the process would start again with the new time period. There are approximately three updates per second in the data tapes. By extracting one per second, very few errors occurred in the display due to errors from telemetry data. Occasionally the telemetry data will drop out due to the aircraft orientation to the receiving antenna. By taking only one sample per second we are almost assured of getting one good data point per second. This information was identified in the data file by the word "position" followed by the time stamp for the data set. This time stamp allows a running clock on the screen during the display sequence.

Once the static and dynamic data were completely filtered, the data was passed through one more filter process to correct the occasional telemetry errors due to aircraft position, antenna position, and signal reflections. This caused some errors in the  $\vec{Z}$  direction cosines which caused errors in the display. To correct this error, the data was filtered one more time to correct the  $\vec{Z}$  direction cosines for aircraft position. The  $\vec{X}$  and  $\vec{Y}$  direction cosines seemed to be correct and were used to restore the  $\vec{Z}$  direction cosines. To correct the  $\vec{Z}$  direction cosines the  $\vec{X}$  and  $\vec{Y}$  directions cosines were used in the following manner:

$$\vec{Z} = \sqrt{1 - (\vec{X}^2 + \vec{Y}^2)}$$

The sign of the  $\vec{Z}$  direction cosine was assumed to be correct and was used for the sign of the regenerated  $\vec{Z}$  direction cosines. Once the database was properly filtered, the main software system was implemented.

## 5.2 Main Module

The main module starts by initializing the graphics workstation to the proper display format. This format can set the display to an NTSC format or use the system's regular 60HZ display format. The Polhemus tracker is initialized to a predetermined state, which sets the desired hemisphere, the output list, and several associated factors. The desired database is then read into memory. The main module then goes into a loop until the user desires to exit the program. This loop consists of several steps:

1. The Polhemus is polled to send back a position report.
2. The objects are moved according to their position and orientation differences in the data record.
3. The mouse is checked to see if there is any input from the user to move into the closest aircraft or move in the direction which they are looking.
4. The keyboard is checked to see if there is any input from the user to control the manner in which the program runs.
5. The time of the display is updated to reflect the progress through the time dependent data record.
6. The Polhemus position report is read and updated to reflect where the user is looking.
7. The view point is updated based on the head movement and the user input.
8. The scene is displayed.
9. The loop is started again.

Each step is discussed in greater detail below.

*5.2.1 Polhemus polling, Polhemus position report* A full discussion of these topics can be found in the Polhemus subunit section.

*5.2.2 Time of data record* The contents of the data record are stored in a sequential time ordered fashion. Each object has a descriptor in the data record which describes the type of object, the object position, and the object orientation vector at a specific time period. The object position and orientation are stored in an array indexed by the aircraft slot position. This slot position is one which was assigned to a particular aircraft while participating in a Red Flag training exercise at Nellis. The aircraft for each slot position is defined to be of a particular type and identified as a member of the red or blue forces. This information is found in a data record only when the participants change. Otherwise there is only position and orientation information with a time stamp at the beginning of each record. This time stamp is displayed on the screen when the program is operating. The display speed can be adjusted by the user for half the display rate, twice the display rate, or the normal display rate. This in effect moves the display at twice the normal clock rate, half the normal clock rate, or at the normal clock rate. The time can also be readjusted by the user to step to a specific time in the data record. By stepping through each time-dependent data record the display is animated.

**5.2.3 Object Movement** Each object is moved based on the information in the data record which relates to the object. The object's position and orientation are specified in three dimensions. As each new time record is acquired the new values are read which relate to the object. These values are compared against the previous values to create a difference which will be used for the display of objects between the times specified in the data record. This gives a linear interpolation to the path which the objects are following. The difference is divided by the number of frames to display between data records to give delta values for position and orientation. The deltas are added to the current object vectors to create the new position and orientation. This value is then used for the display of the object in the next frame.

**5.2.4 User Input** The user has two choices for program control. Both a mouse and a keyboard can be used for input to control program flow. The set of commands available to the user is the different for each input device. If the mouse is used for input the following options are available:

- Move forward at a speed approximately equal to the aircraft.
- Move forward at a speed about ten times the aircraft.
- Move backward at a speed about ten times the aircraft.
- Toggle into or out of the closest aircraft cockpit.

There are several options available to the user for input from the keyboard. These options are:

- Alter the display rate to double the normal speed.
- Alter the display rate to half the normal speed.
- Alter the display rate to return to normal speed.
- Go to a specified time.
- Enter a specific cockpit.
- End the program.

The preferable method would have been to allow the user complete freedom from the keyboard but this was not possible with input devices used in the current system.

**5.2.5 Determining the viewpoint** Several inputs are used to determine where the user is looking in the situation. Initially the user is placed at an arbitrary point on the Nellis range. The orientation of the user is based directly from the input of the Polhemus tracker. If the user selects an object then the user is placed at the location of the object. The user continues to move with the object until he exits the object. While moving with an object the user's position is not dependent on the position of the helmet but the position of the object. The view displayed to the user is dependent upon input from the Polhemus tracker and the orientation of the object. To receive orientation information from the Polhemus tracker a simple one line command is issued. However, several steps need to be accomplished prior to issuing a command to the Polhemus Tracker.

### **5.3 Polhemus Unit**

The Polhemus unit takes care of the necessary interface to the tracking device on the head-mounted display. There are several subunits in this unit. One sub unit is used to initialize the Polhemus tracker to the desired operating parameters. The second sub unit is responsible for polling the Polhemus and the last sub unit is responsible for retrieving the position information from the Polhemus.

**5.3.1 Initialization** The Polhemus tracker is initialized to return very few parameters. These parameters are necessary for the calculation of the viewing parameters for a proper display based on where the user is looking. To initialize the Polhemus tracker several steps are taken:

1. Open the communications channel between the workstation and the Polhemus.
2. Reset the Polhemus to the factory set defaults.
3. Redefine the coordinate system used by the Polhemus tracker.
4. Specify non-continuous ASCII output of data with the output given in inches.
5. Define the minimum and maximum X, Y, and Z values.
6. Define the current hemisphere to have the zenith along the Z-Axis.

Many of these steps are used to insure that EEPROM in the Polhemus tracker has not been reset by another user. Once the Polhemus tracker is properly initialized, the orientation information can be obtained.

**5.3.2 Polling the Polhemus Tracker** The Polhemus tracker is polled once through the main unit inner loop. This is accomplished by sending a command to the Polhemus to send the data to the main unit. This process requires only a single character is sent to the Polhemus tracker.

**5.3.3 Retrieving Polhemus Tracker Information** Information is retrieved from the Polhemus tracker after several program steps have been executed by other software routines. This delay is necessary to allow the Polhemus tracker time to respond to the position request and the delay associated with transmission of data through the communications channel.

#### **5.4 Data Handling**

Data is handled in several different ways. Initially data is filtered and adjusted to meet the data format required by the data display routines. The data is then stored in a sequential file for use by the sub unit to traverse the database. Another sub unit is present to allow the user to move to a specified time in the sequence of events.

**5.4.1 Traversing the Data** The data is traversed in a sequential manner while the main inner loop is executing. If a user decides to change to a format other than normal speed only the number of frames displayed between data records needs to be changed. At a normal traversal speed the present and next data record are compared to calculate the differences between aircraft position and orientation. The change in orientation and position are divided up into smaller intervals. This interval is used to change the position and orientation of the object. When the object is displayed the interval is added to the previous position and orientation vectors to reposition the object in a new location.

If the object is moving at half speed, twice as many frames are displayed between reading data records. On the other hand if the user desires a double speed, one half as

many frames are displayed between reading the data records.

**5.4.2 Move to a Specified Time** When the user desires to move to a specific time, he must first stop the sequence of events. Then the user can select a specified time in an hours:minutes:seconds format. The module then calculates the nearest time step and moves to the appropriate data record. Following this the sequences of events proceeds from the selected time forward. If the user inserts an invalid time the display time is set to the nearest display time. For example if the user set the time before the beginning of the display sequence the display sequence would start at the beginning of the data base.

Once the data records are manipulated and the object parameters are updated the objects are put in their new locations. Now the objects will be displayed based on the last data update.

## **5.5 Display Unit**

The display unit is broken into three parts. The first sub unit is called on program entry to initialize all the objects which will be used in the display sequence. The second sub unit is responsible for moving the objects based on the deltas calculated by the routine which traverses the data records. The third sub unit is responsible for the display of objects from the user's present viewpoint.

**5.5.1 Initialize Objects** The objects are usually defined once in the first part of the data record. If the objects are redefined later, another data set is created. This new data set will not be visible until the proper time is reached in the display sequence. The objects are initialized to a specific aircraft and color at the time of initialization. These objects are defined from one of the models in the database. These objects are not associated with a specific position until the display sequence is started.

**5.5.2 Object Movement** When the time arrives for a new image the objects are moved by changing their position and orientation vectors. This is done by inserting the new values onto the object stack. If the object is within the restrictions of the range the object

is displayed. Otherwise the object is ignored. This is very helpful in eliminating objects which are on their way to the range. This allows display of only the active participants.

*5.5.3 Object Display* After all the objects positions have been updated the display module checks the user's position. If the user is not in one of the objects the information from the Polhemus is used to determine where the user is looking. On the other hand, if the user is in an object the position vector of the object is used to determine where the user is located. The position vector of the object and the position vector of the Polhemus tracker are used to determine where the user is looking. This allows the display to tilt as if the user were seated directly in the cockpit.

The display unit continues to get the next data record and calculate the new viewing parameters indefinitely. This allows the user to view the engagement continuously without leaving the virtual environment.

## *5.6 Summary*

The implementation of the software system was very successful due to the planning prior to coding. The problems encountered in filtering data were overcome by using multiple levels of the filter routine. These multiple levels allowed the reconstruction of missing orientation information by using the extra information in direction cosines. All the filtering of data was successfully accomplished with little interaction of the user necessary. The Polhemus tracking system provided the necessary orientation information with minimal time delay from the initial data request to the reception of orientation information. The user interface allows the desired freedom from the keyboard. This capability permits the user to remain within the virtual world environment and still control the software system. Finally, the display routines were flexible enough to allow various display speeds to the user.

## *VI. Conclusions and Recommendations*

This chapter is the summation of the thesis effort. In this chapter are conclusions on the thesis followed by suggestions for future enhancements of the software or other questions which have arisen due to this thesis effort.

### *6.1 Conclusions*

There are several areas of this thesis which were satisfying and other areas which were not very productive. Overall, the thesis has given some insight into the display of time dependent data with a head-mounted display. The positive results of the thesis will be described first followed by the results which were not so promising.

The filtering of the Red Flag data tapes made the display of a time dependent scenario more believable than data generated by other means. The Polhemus tracking device allowed the user to be free from the computer. With the addition of a mouse for command input, the user may remain in the virtual world and still control the system. The user interface appears to be quite easy to use. Several users quickly developed the skills necessary to maneuver around the virtual world after a short instructional session. These were the promising results of the thesis. There were a few results which were discouraging though.

Initially, the users had a tendency to remain stationary and not take advantage of the virtual world system. The users were hesitant to look around and move around within the virtual world. This may have been caused by the cables necessary to provide the image to the head mounted display or the lack of familiarity with head-mounted displays. Due to the size of the Red Flag range, models had to be sized to 10 times their actual size. This had the disadvantage of misrepresenting the actual distances between aircraft when they are flying. Options input from the keyboard were restrictive and often required assistance of another person. An alternative method of user input is necessary to free the user entirely from the computer keyboard. Based on the results, there are several recommendations for future work in this thesis area.



## **6.2 Recommendations**

The software system appears to validate the proof of concept, yet there are several areas of the system which could use improvement. A few of the areas which should be considered are:

- Incorporating another device to totally free the user from the keyboard of the computer. Consider the adaptation of a joystick or implementing a menu system where an item is selected by the movement of the user's head and a mouse button.
- Improve the man-machine interface to make the system easier to use, even by the novice.
- Incorporate adaptive refinement techniques to add shading and a solution to the hidden surface problem when a user pauses the display.
- Make the system more general to handle different models and present different environments which rely on time dependent data.
- Transfer the software system to a computer with faster processing capabilities. This will allow an increase in the display rate. Since the Iris was computationally bound with this application.

Some other areas need to be researched using the head-mounted display. A study into the effects of stereopsis should be conducted to see if there are any improvements from adapting this to the current display system. The idea of whether such a device would actually be a valid training resource needs to be considered. The undergraduate pilot training program may profit from this device when training new pilots. Battle staff management may get greater assistance when viewing a battle plan. The use of time dependent data with a head-mounted display should be further researched for other applications.

## Appendix A. Calculation of the Viewing Parameters

To look at something in the real world we simply turn our head to look at an object. When using computer graphics the process is not as simple. Instead, a series of operations using matrix multiplication are required (11:348-351). When calculating these values we are attempting to change the object positions from a world coordinate system to an eye coordinate system in one operation. The matrix created by these calculations can be multiplied by our world coordinates to get our eye coordinates.

The world coordinate system is a right handed system and the eye coordinate system is based on a left handed coordinate system. We start this transformation process by first defining in world coordinates where our eye is located and a point P where the eye is looking.

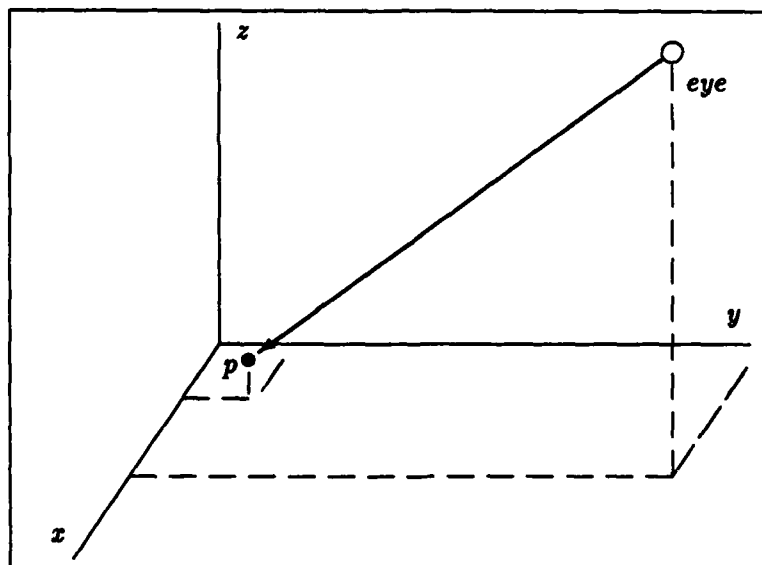


Figure 4. Determination of Line of Sight

These points will determine our line of sight. These points are taken from the position data which is passed from the Polhemus tracker.

The Polhemus tracker is a system used to measure position and orientation in a confined area. This is accomplished with a transmitter (the source) and a receiver (the sensor). The source emits a low frequency magnetic field which is detected by the sensor.

After the information is detected by the sensor, a microprocessor calculates the sensor position and orientation based on the data received from the sensor. The Polhemus tracker allows the return of several different types of orientation and position data (12). For this thesis the X, Y, and Z coordinates of the sensor, the direction cosines of the X axis, and the twist angle of the sensor were used. The coordinates and orientation of the sensor are relative to the coordinate system of the source. The eye position becomes the X, Y, and Z values from the Polhemus tracker and may be multiplied by a scaling factor. The line of sight given from the eye position and the direction cosines of the source, give the direction of the point we are looking at. The points  $p$  and  $eye$  give our line of sight. The twist angle is used to determine the rotation about the Z axis in our eye coordinate system. Our line of sight is now determined and we can determine the parameters necessary to transform our objects from world coordinates to eye coordinates. The initial matrix we will start with is an identity matrix.

By concatenating several simple matrices, we can form a simple matrix that will convert from world coordinates to eye coordinates in one multiplication instead of several discrete steps.

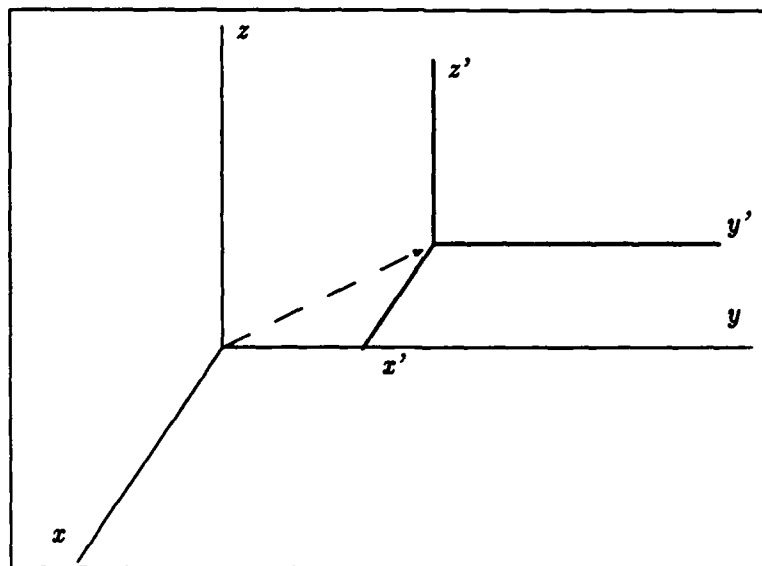


Figure 5. Translation to Eye Coordinate System

Initially the eye position origin is translated to the origin of the world coordinate

system. This is done by inserting the X, Y, and Z positions into the matrix after they have been multiplied by minus one to move from the origin to the point our eye is looking.

$$\begin{vmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ -X & -Y & -Z & 1.0 \end{vmatrix} \quad (1)$$

The new coordinate system is now rotated 90 degrees about the X axis. The matrix is multiplied by the previous matrix to give:

$$\begin{vmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -X & -Z & Y & 1.0 \end{vmatrix} \quad (2)$$

Pictorially we have done the following to our coordinates:

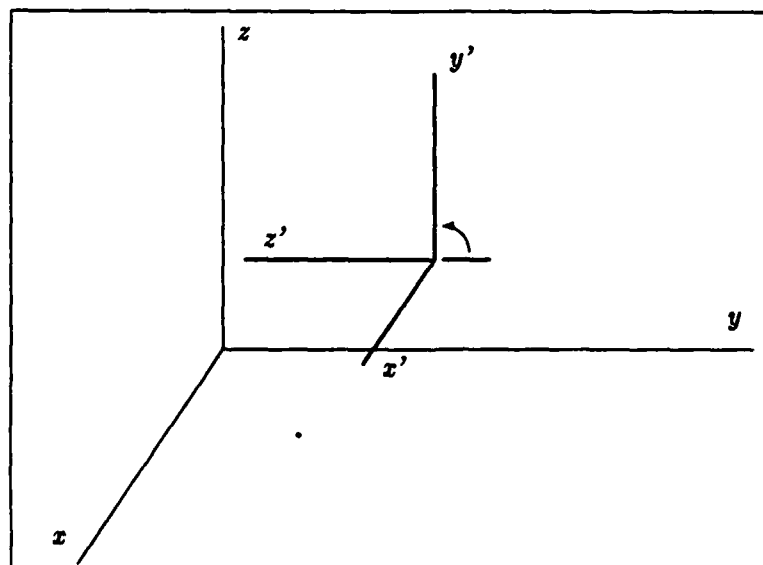


Figure 6. Ninety Degree Rotation about the X Axis

Now we rotate about the Y' axis by the so the point we are looking at will lie on the Y'-Z' plane. This rotation is done by the following matrix:

$$\begin{vmatrix} \cos \theta & 0.0 & -\sin \theta & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ \sin \theta & 0.0 & \cos \theta & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{vmatrix} \quad (3)$$

This operation is illustrated in Figure 7.

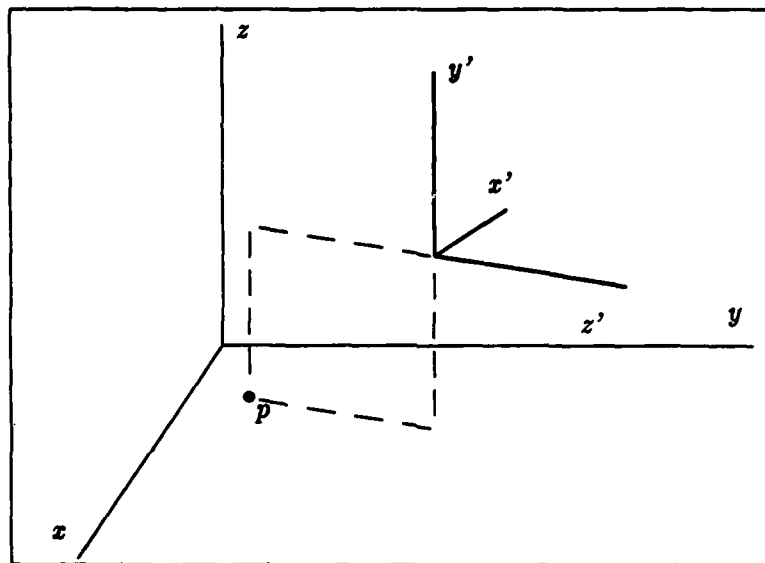


Figure 7. Rotation about the Y' Axis

The resulting transformation matrix is now represented as in equation 4.

$$\begin{vmatrix} \cos \theta & 0.0 & \sin \theta & 0.0 \\ -\sin \theta & 0.0 & -\cos \theta & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -X \cos \theta + Y \sin \theta & -Z & X \sin \theta + Y \cos \theta & 1.0 \end{vmatrix} \quad (4)$$

Rotation is performed about the X' axis so the point we are looking at will lie on the Z' axis as illustrated in Figure 8.

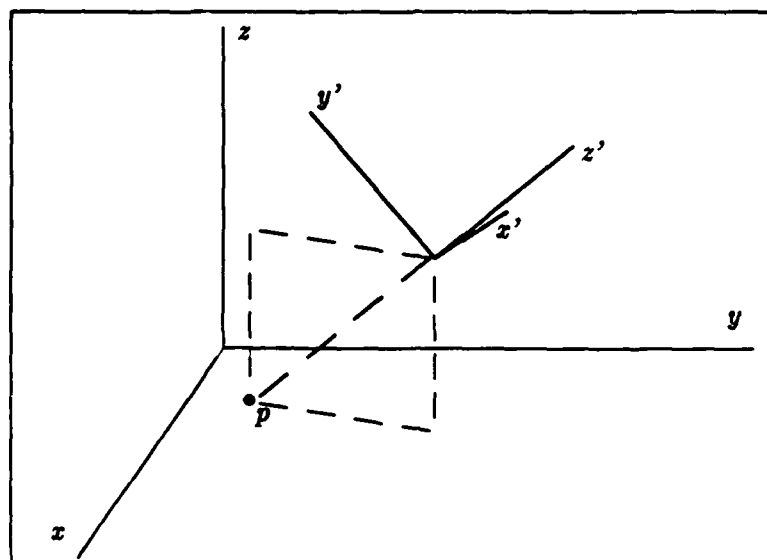


Figure 8. Rotation about the X' Axis

The matrix to do this operation is:

$$\begin{vmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & \cos \varphi & -\sin \varphi & 0.0 \\ 0.0 & \sin \varphi & \cos \varphi & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{vmatrix} \quad (5)$$

Following this operation our transformation matrix will look as follows:

$$\begin{vmatrix} \cos \theta & -\sin \theta \sin \varphi & \sin \theta \cos \varphi & 0.0 \\ -\sin \theta & -\cos \theta \sin \varphi & -\cos \theta \cos \varphi & 0.0 \\ 0.0 & \cos \varphi & -\sin \varphi & 0.0 \\ m_{4,1} & m_{4,2} & m_{4,3} & 1.0 \end{vmatrix} \quad (6)$$

Where:

$$m_{4,1} = -X \cos \theta + Y \sin \theta$$

$$m_{4,2} = -Z \cos \varphi + X \sin \theta \sin \varphi + Y \cos \theta \sin \varphi$$

$$m_{4,3} = Z \sin \varphi + X \sin \theta \cos \varphi + Y \cos \theta \cos \varphi$$

The  $Z'$  axis is now looking away from the object. So the sense of the  $Z'$  axis is reversed. This accomplishes two things. First, the coordinate system is now changed from a right handed system to the desired left handed system of our eye coordinate system. Second, the user is now looking at the object as illustrated in Figure 9.

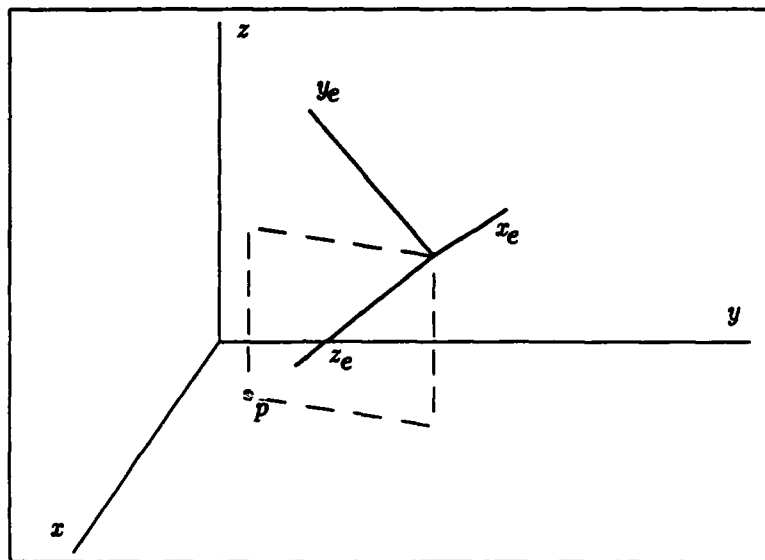


Figure 9. Reversal of the  $Z'$  Axis

All that is required for matrix operations is to multiply element 3,3 by -1.0 to reverse the sense of the direction, resulting in:

$$\begin{vmatrix} \cos \theta & -\sin \theta \sin \varphi & \sin \theta \cos \varphi & 0.0 \\ -\sin \theta & -\cos \theta \sin \varphi & -\cos \theta \cos \varphi & 0.0 \\ 0.0 & \cos \varphi & \sin \varphi & 0.0 \\ m_{4,1} & m_{4,2} & m_{4,3} & 1.0 \end{vmatrix} \quad (7)$$

The last step is to rotate around the  $Z$  axis by the amount of twist which was provided by the Polhemus tracker.

This will require the previous matrix to be multiplied by:

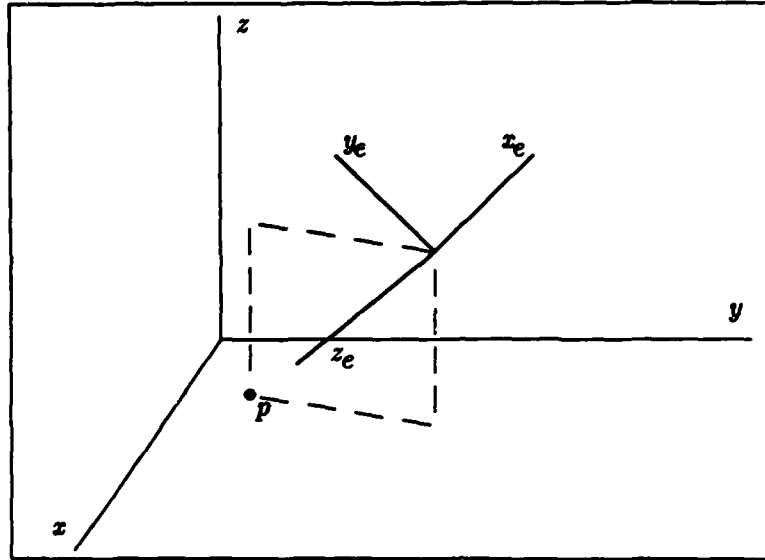


Figure 10. Rotation about the Z' Axis

$$\begin{vmatrix} \cos \epsilon & -\sin \epsilon & 0.0 & 0.0 \\ \sin \epsilon & \cos \epsilon & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{vmatrix} \quad (8)$$

The matrix formed by these operations will transform world coordinates into eye coordinates. This will allow all the points in the world coordinate system to be multiplied by the matrix formed, transforming the world coordinates into eye coordinates. This matrix is:

$$\begin{vmatrix} \cos \theta \cos \epsilon - \sin \theta \sin \varphi \sin \epsilon & -\cos \theta \sin \epsilon - \sin \theta \sin \varphi \cos \epsilon & \sin \theta \cos \varphi & 0.0 \\ -\sin \theta \cos \epsilon - \cos \theta \sin \varphi \sin \epsilon & \sin \theta \sin \epsilon - \cos \theta \sin \varphi \cos \epsilon & -\cos \theta \cos \varphi & 0.0 \\ \cos \varphi \sin \epsilon & \cos \varphi \cos \epsilon & -\sin \varphi & 0.0 \\ m_{4,1} & m_{4,2} & m_{4,3} & 1.0 \end{vmatrix} \quad (9)$$

Where:

$$\begin{aligned} m_{4,1} = & -X \cos \theta \cos \epsilon + Y \sin \theta \cos \epsilon \\ & - Z \cos \varphi \sin \epsilon + X \sin \theta \sin \varphi \sin \epsilon + Y \cos \theta \sin \varphi \sin \epsilon \end{aligned}$$



$$\begin{aligned}
m_{4,2} &= X \cos \theta \sin \epsilon - Y \sin \theta \sin \epsilon \\
&\quad - Z \cos \varphi \cos \epsilon + X \sin \theta \sin \varphi \cos \epsilon + Y \cos \theta \sin \varphi \cos \epsilon \\
m_{4,3} &= Z \sin \varphi + X \sin \theta \cos \varphi + Y \cos \theta \cos \varphi
\end{aligned}$$

This will provide a regular transformation from a point in the world coordinate system to a point in a viewing coordinate system. By using only the orientation information from the Polhemus tracker the transformation matrix can be simplified by eliminating the twist angle multiplication. Another multiplication which can be eliminated is the reversal in the direction of the Z Axis. This is automatically done by the Iris when calculating the perspective transformation! The Iris handles the matrices a bit differently than the usual matrix multiplication. The Iris premultiplies the matrices instead of the normal postmultiply use in matrix multiplication. This allows the user to specify where they are looking before specifying the modeling transformations to be applied. By using such a scheme several objects can use the same transformation matrix to establish their position in viewing coordinates after the modeling transform has been applied to each model.

## Appendix B. *Guide for Filtering Data*

The process of filtering data is not that difficult once the format of the data is understood. One merely has to extract the necessary parts and discard the rest of the data. To start filtering the data, the data tapes were down loaded using the "dd" facility on Unix based machines. The input parameters for this were:

- *if= tape unit*
- *of= filename*
- *bs= 65536*

These parameters led to very few errors in the data file. However the resulting data file was too large to use. So the file was pared down with a filter program "filterdata." To use filterdata the user must specify the input and output files on the command line (i.e. *filterdata infile outfile*). The output from this program was usable for the display of models and their movement. Occasionally there are data dropouts with the Z direction cosines. To cure this run ReFilter. This will reconstruct the Z direction cosines from the X and Y direction cosines. Use the command "*ReFilter infile > outfile.*" After this final filter process the data is ready for use.

## Appendix C. *User's Manual*

The software system developed for using the time-dependent data and the head mounted display is easy to use. To run the system the user should perform the following steps.

1. Install the proper connections to the helmet. There are two of these. The first plugs into the connection at the back of the helmet. The second connection is a velcro fastener which goes on the velcro patch on the top of the helmet.
2. Remove the mouse from the Iris keyboard. Connect the mouse to the remaining nine pin connector.
3. Turn the power switch located on the power strip on the left hand side of the equipment shelf.
4. Turn on the Lyon-Lamb NTSC converter. This is the gray box located to the right of the Iris monitor.
5. Change to the "demo" directory and run the program "hmd." If you want to run other than the demo data set use "tracker *filename* [-n] [-bl]." This will give the system the use of the named file. The "-n" option will set the monitor to NTSC mode. The "-bl" option will set the output to a blue background.

Once this is done the system will load the data set. Be patient; this may take a while depending on the size of the data set. To use the system put on the helmet and grab the mouse. While away from the keyboard the following commands are available:

1. To toggle inside or outside the cockpit nearest your position use the left mouse button.
2. To proceed at a normal rate in the direction you are looking press the right mouse button. Continue to press the mouse button to continue moving.
3. To proceed at a fast rate in the direction you are looking press and hold the right and middle mouse button simultaneously. This mode is normally used to catch up to an aircraft you are looking at, or to move to an area of the range you are interested in seeing.

4. To proceed at a fast rate in the direction opposite to which you are looking depress and hold the middle mouse button.
5. To see a different part of the scene turn your head.

Additional commands are available to the user from the keyboard are:

1. 'q' to quit the program.
2. 'c' to enter a specific cockpit. The system will prompt you for the cockpit number. Type in the cockpit number followed by return.
3. 't' to set display time to a specific time. The system will prompt you for the desired time. Enter the time as hh:mm:ss. Follow the time entry with a return
4. 'd' to double the speed of display routine.
5. 'h' to halve the speed of the display routine.
6. 'n' to set the display speed to normal.

That is all there is to running the system.

### Bibliography

1. Bergman, Larry and others. "Image Rendering By Adaptive Refinement," *Computer Graphics*, 20: 29-38 (August 1986)
2. Bishop, Gary and David M. Weimer. "Fast Phong Shading," *Computer Graphics*, 20: 103-106 (August 1986)
3. Brooks, Frederick P. Jr. "Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings" *Proceedings of the ACM 1986 Workshop on Interactive 3D Graphics*. 9-21. New York: ACM Press, 1986
4. Cubic Defense Systems, *Computer Program Performance Specification for the CCS*, Documentation, Cubic Defense Systems RFMDS, Nellis AFB NV, 89191
5. Fisher, S.S. and others. "Virtual Environment Display System" *Proceedings of the ACM 1986 Workshop on Interactive 3D Graphics*. 77-87. New York: ACM Press, 1986
6. Foley James D., and Andries Van Dam. *Fundamentals of Interactive Computer Graphics*. Philippines: Addison-Wesley Publishing Company, 1982.
7. Foley James D. and others. "The Human Factors of Computer Graphics Interaction Techniques." *IEEE Computer Graphics and Applications* 4: 13-18 (11 November 1984)
8. Fuchs, Henry and others. "Near Real-Time Shaded Display of Rigid Objects," *Computer Graphics*, 17: 65-69 (July 1983)
9. Hayes, John P. *Computer Architecture and Organization*. New York: McGraw-Hill Book Company, 1978.
10. Mosher, Charles E. Jr. and others. "The Virtual Simulator" *Proceedings of the ACM 1986 Workshop on Interactive 3D Graphics*. 37-42. New York: ACM Press, 1986
11. Neuman, Willam M. and Robert F. Sproull. *Principles of Interactive Computer Graphics*. New York: McGraw Hill Book Company, 1979.
12. Polhemus Navigation Sciences Division. *3Space User's Manual*. McDonnell Douglas Corporation, Colchester, Vermont 05446
13. Stone, Harold S. *High-Performance Computer Architecture* Reading, Massachusetts: Addison-Wesley Publishing Company, 1987.
14. Zyda, Michael J. and others *An Inexpensive Real-Time Interactive Three-Dimensional Flight Simulation System*. Masters Thesis. Naval Postgraduate School Monterey CA, (AD A184340)

Vita

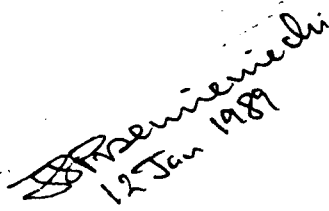
Captain Gary K. Lorimor [REDACTED] He graduated from Corvallis High School in Corvallis, Oregon in 1975 and entered Oregon State University in Corvallis, Oregon. He graduated in 1980 with a Bachelor of Science in electrical engineering specializing in computer engineering. Captain Lorimor entered undergraduate pilot training (UPT) in June of 1980. Following graduation from UPT, Captain Lorimor served three and a half years at Vance AFB, Oklahoma as a T-37 instructor pilot. He then served two years at Norton AFB, California as a C-141 instructor pilot before entering the School of Engineering, Air Force Institute of Technology in June 1987. [REDACTED]  
[REDACTED]  
[REDACTED]  
[REDACTED]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) <b>AFIT/GE/ENG/88D-22</b>			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION <b>School of Engineering</b>		6b. OFFICE SYMBOL (If applicable) <b>AFIT/ENG</b>		7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) <b>Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583</b>			7b. ADDRESS (City, State, and ZIP Code)	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION <b>AAMRL</b>		8b. OFFICE SYMBOL (If applicable) <b>HEA</b>		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) <b>Wright-Patterson AFB OH 45433</b>			10. SOURCE OF FUNDING NUMBERS	
			PROGRAM ELEMENT NO.	TASK NO.
			PROJECT NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) <b>REAL-TIME DISPLAY OF TIME DEPENDENT DATA USING A HEAD-MOUNTED DISPLAY</b>				
12. PERSONAL AUTHOR(S) <b>Gary K. Lorimor, B.S., Capt, USAF</b>				
13a. TYPE OF REPORT <b>MS thesis</b>		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) <b>1988 December</b>
15. PAGE COUNT <b>61</b>				
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) <b>Helmet mounted displays Computer graphics Position finding</b>	
FIELD	GROUP	SUB-GROUP		
<b>25</b>	<b>03</b>			
<b>12</b>	<b>07</b>			
19. ABSTRACT (Continue on reverse if necessary and identify by block number)  <b>Thesis Advisor: Phil Amburn, Major, USAF Professor of Computer Systems</b>				
<div style="text-align: right;">   12 Jan 1989 </div>				
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>	
22a. NAME OF RESPONSIBLE INDIVIDUAL <b>Phil Amburn, Major, USAF</b>			22b. TELEPHONE (Include Area Code) <b>(513) 255-3576</b>	
			22c. OFFICE SYMBOL <b>AFIT/ENG</b>	

## ABSTRACT

→ The purpose of this investigation was the development of a software system to integrate time-dependent data with a three-dimensional virtual environment. Red Flag data tapes were used as a source of time-dependent data. To project this into a virtual environment a head-mounted display was used. The software development was done on a Silicon Graphics Iris 3130 workstation. The design of the software system allows the user to be free of the keyboard for most situations. The position of the user's head was determined using a Polhemus 3-space tracker. The user could exercise basic control of the system with input from mouse buttons. The user was allowed to move to any position in the environment by looking in the direction of desired travel. The user was also given the option to ride in any cockpit which was in the environment. This allowed the user to see the environment from a pilots perspective at a specific point and time in the display sequence. Minimal machine dependent routines were used in the development of the software system. The software system was developed using the C programming language. The results of this effort were encouraging. There appears to be further possibilities of use for a system of this type in the training arena. The update rate of this system is about 10 frames per second. There appears to be several open questions concerning the benefits of such a system in the training environment.

Theses. (SDW) →